Clustering - Advanced Concepts

Krzysztof Wolny

28 stycznia 2025

Spis treści

1 Wprowadzenie **2**

2 Analiza danych **3**

3 DBSCAN **5**
3.1 Opis algorytmu 5
3.2 Wyniki 5

4 Genie Method **8**
4.1 Opis algorytmu 8
4.2 Wyniki 8

5 Podsumowanie wyników **9**

1 Wprowadzenie

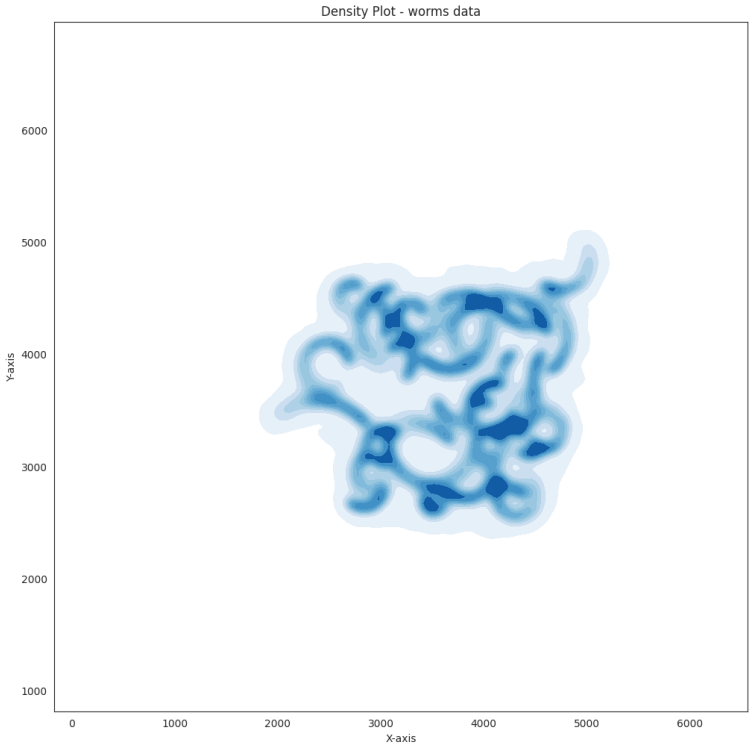
Celem raportu jest klasteryzacja danych z zestawu `worms_2`, dostępnego pod adresem: `https://github.com/gagolews/clustering-data-v1/tree/master/sipu`. W ramach analizy wykonano następujące kroki:

1. Przeprowadzono klasteryzację danych przy użyciu metody Genie oraz metody DBSCAN, wraz z analizą doboru optymalnych parametrów.
2. Wyniki klasteryzacji porównano z rozwiązaniem referencyjnym, dostępnym dla tego zbioru danych.
3. Opracowano wizualizacje wyników, stosując odpowiednie techniki wizualizacji danych.
4. Dokonano oceny jakości uzyskanych rozwiązań, uwzględniając ich zgodność z podanym rozwiązaniem oraz interpretację wyników.

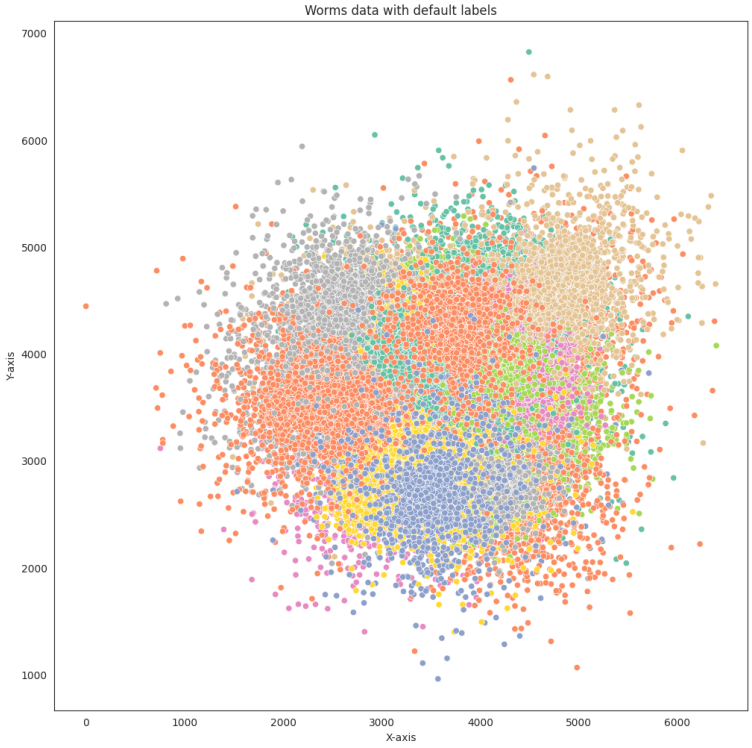
2 Analiza danych

Zbiór danych worms jest zbiorem dwuwymiarowym. Posiada 105 600 wierszy. Zakres danych wymiarów wynosi (0, 6397) oraz (958, 6821.8). Na wykresie [1] widać gęstość danych. Widzimy, że są skupiska gęstszych miejsc oraz trochę rzadsze połączenia pomiędzy tymi skupiskami. Dookoła są dane o mniejszej gęstości.

W zadaniu zostało podane przykładowe rozwiązanie dzielące zbiór danych na 35 klastrów. Podział danych na wykresie jest przedstawiony na wykresie [2]. Największy klaster posiada 5880 punktów, a najmniejszy 768. Możemy zaobserwować, że klastry w rozwiązaniu się na siebie nakładają.



Rysunek 1: Gęstość danych worms.



Rysunek 2: Dane worms z podaną klasteryzacją.

3 DBSCAN

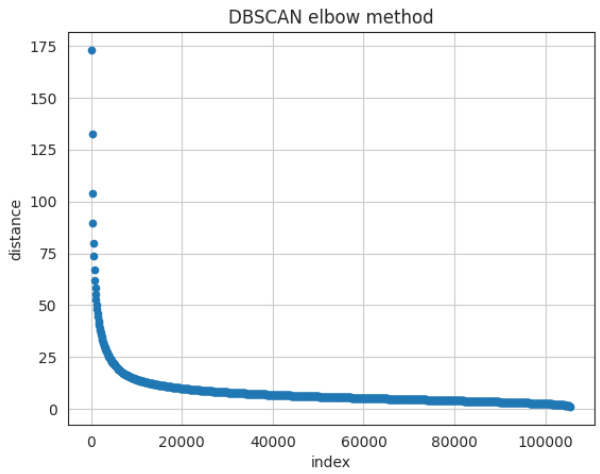
3.1 Opis algorytmu

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) to algorytm klasteryzacji, który grupuje dane na podstawie ich gęstości. Działa, identyfikując punkty rdzeniowe, czyli takie, które mają co najmniej określoną liczbę sąsiadów (parametr `MinPts`) w promieniu `eps`. Punkty sąsiadujące z punktami rdzeniowymi są przypisywane do tego samego klastra co najbliższy punkt rdzeniowy. Punkty, które nie mają sąsiada, są oznaczane jako szum (punkty odstające). Kluczowe dla jakości wyników algorytmu jest odpowiednie dobranie parametrów `eps` i `MinPts`.

3.2 Wyniki

Do znalezienia odpowiednich parametrów `eps` oraz `MinPts` posłużyłem się metodą 'łokcia'[1][2]. Metoda służy do ustalenia optymalnej wartości parametru `eps` w algorytmie DBSCAN dla określonego `MinPts`. Dla danych dwuwymiarowych na początku ustala się wartość `MinPts` $= 4$. Następnie dla każdego punktu oblicza się odległości do `MinPts`-tego najbliższego sąsiada, wykorzystując metodę k-nearest neighbors, gdzie $k =$ `MinPts`. Kolejnym krokiem jest posortowanie obliczonych odległości w porządku malejącym. Na podstawie tych odległości tworzy się wykres, gdzie oś x przedstawia punkty danych, a oś y odpowiada odległościom. Punkt 'łokcia' na wykresie wskazuje optymalną wartość `eps`. Jest on wybierany w sposób subiektywny. Dzięki temu możliwy jest dobór odpowiedniego parametru `eps`.

Przy danych worms ustaliłem `MinPts` równe 4 oraz patrzyłem na rezultaty algorytmu. Stworzyłem wykres posługując się metodą 'łokcia' [3]. Według wykresu 'łokieć' widać było przy wartościach 10-25. Wybrałem kilka przykadowych `eps` i sprawdziłem rezultaty. Dla mniejszych `eps` otrzymywałem dużą ilość mniejszych klastrów. Przy zwiększaniu `eps` tworzył się jeden duży klaster oraz dużo mniejszych na obrzeżach. Wyniki dla `eps` $= 24$ oraz `eps` $= 10$ widać na wykresie [4] oraz w tabeli [3.2].



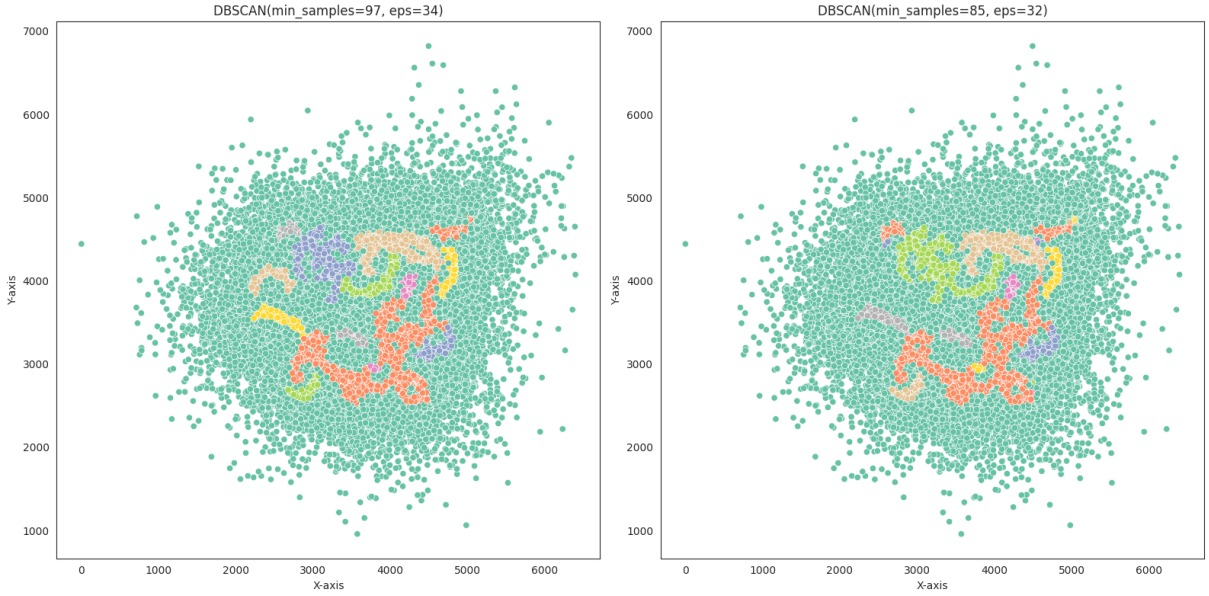
Rysunek 3: Wykres metody 'łokcia' dla algorytmu DBSCAN dla `MinPts` $= 4$.

Metoda	Ilość klastrów	Outlier share
DBSCAN(`MinPts` $= 4$, `eps` $= 24$)	268	3%
DBSCAN(`MinPts` $= 4$, `eps` $= 10$)	1258	13.5%

Tabela 1: Tabela z porównaniem DBSCAN dla `MinPts` $= 4$.

Rysunek 4: Klasteryzacja danych przy użyciu metody DBSCAN(`MinPts` = 4, `eps` = 24) oraz DBSCAN(`MinPts` = 4, `eps` = 10).

W związku z niezadowalającymi wynikami spróbowałem zwiększyć parametr `MinPts`. Sprawdziłem kilka przykładowych `MinPts`. Stworzyłem do nich wykresy metody 'łokcia', aby łatwiej było dobierać dla nich `eps`. Po analizie kilku wyników zdecydowałem się zrobić grid search dla parametrów. Szukałem najlepszej opcji dla `MinPts` od 80 do 100 oraz `eps` od 27 do 40 z krokiem równym 1. Łącznie wynosiło to 260 par. Do sprawdzenia jakości klasteryzacji mierzyłem ilość klastrów dla każdego zestawu parametrów oraz procent punktów odstających. Postanowiłem również obliczyć Adjusted Rand Score w porównaniu z podanym w zadaniu rozwiązaniem. Najlepsze wyniki pod względem Adjusted Rand Score otrzymałem dla zestawów `MinPts` = 97, `eps` = 34 oraz `MinPts` = 85, `eps` = 32. Wykresy z wynikami widać na wykresie[5]. Otrzymałem bardzo dużą ilość punktów odstających - aż 30%. Klastry są tworzone przy gęstszych miejscach zbiory danych. Ilość klastrów jest równa odpowiednio 18 oraz 23, czyli mniej niż w podanym rozwiązaniu.



Rysunek 5: Klasteryzacja danych przy użyciu metody DBSCAN(`MinPts`=97, `eps`=34) oraz DBSCAN(`MinPts`=85, `eps`=32).

Podsumowując, bardzo problematyczne w tym problemie jest wybieranie parametrów dla metody DBSCAN. W zależności od dobranych parametrów tworzy nam się, albo jeden duży klaster, albo mniejsze klastry z dużą ilością punktów odstających. Sprawia to, że użycie metody DBSCAN nie jest optymalnym rozwiązaniem dla problemu.

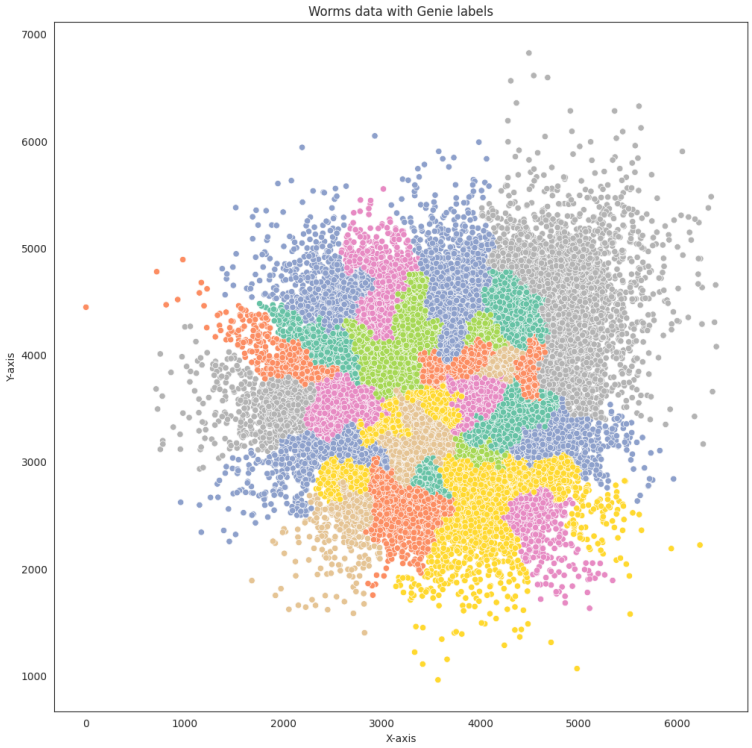
4 Genie Method

4.1 Opis algorytmu

Przy metodzie Genie[3] na początku każdy punkt danych traktowany jest jako oddzielny klaster. Następnie algorytm iteracyjnie łączy najbliższe klastry. Gdy algorytm zauważy, że jeden z klastrów staje się znacznie mniejszy od pozostałych, łączy go z jego najbliższymi sąsiadami, aby uniknąć tworzenia niewielkich i odizolowanych grup.

4.2 Wyniki

Użyłem metody Genie, aby podzielić dane na 35 klastry. W tym przypadku otrzymaliśmy bardzo sensowną klasteryzację danych. Rezultat widać na wykresie [6]. W metodzie Genie wszystkie punkty są przypisywane do klastra, w związku z czym nie mamy outlierów.



Rysunek 6: Klasteryzacja danych przy użyciu metody Genie.

5 Podsumowanie wyników

Do porównania wyników z podanym na początku rozwiązaniem użyłem Adjusted Rand Score porównując klastry otrzymane przez algorytmy Genie i DBSCAN oraz zmierzyłem Silhouette Score. Zmierzyłem również czas wykonywania algorytmu biorąc średnią z 10 pomiarów. Wyniki są przedstawione w tabeli[5]. Możemy z niej wyczytać, że metoda Genie osiągnęła lepsze wyniki zarówno patrząc na Adjusted Rand Score jak i dla Silhouette Score. Silhouette score dla metody Genie jest również lepszy od rozwiązania podanego w zadaniu, który wynosi -0.02. Wskazuje to, że metoda Genie lepiej separuje klastry oraz zapewnia ich większą spójność. Metody DBSCAN mają dużą ilość outlierów. Zaletą jest ich szybkość obliczeń. Porównując wyniki 'na oko' również widać, że najlepsze rozwiązanie daje metoda Genie.

Metoda	Ilość klastrów	Adjusted Rand Score	Time[s]	Silhouette score	Outlier share
DBSCAN(MinPts=97, eps=34)	18	0.1	1.19	-0.23	30%
DBSCAN(MinPts=85, eps=32)	23	0.1	1.16	-0.27	30%
Genie(NrClusters=35)	35	0.38	1.64	0.2	0

Tabela 2: Tabela z porównaniem obu metod.

Literatura

[1] Aggarwal, Charu C. *Data Mining: The Textbook*, 2015, Springer Publishing Company

[2] Ester, Martin and Kriegel, Hans-Peter and Sander, Jörg and Xu, Xiaowei *A density-based algorithm for discovering clusters in large spatial databases with noise*, 1996, AAAI Press

[3] Gagolewski M., Bartoszuk M., Cena A. *Genie: A new, fast, and outlier-resistant hierarchical clustering algorithm*, Information Sciences 363, 8–23, 2016, DOI:10.1016/j.ins.2016.05.003, URL:`https://genieclust.gagolewski.com/`